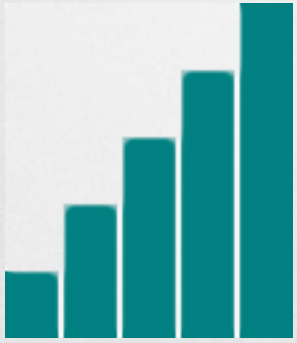
El Poder de la Función data( )

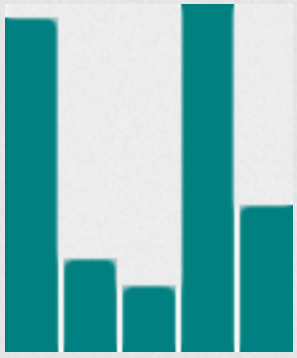
Lo último que hicimos fue un diagrama de barras simple, dibujado utilizando etiquetas `div` y un conjunto de datos sencillo.

```
var dataset = [ 5, 10, 15, 20, 25 ];
```



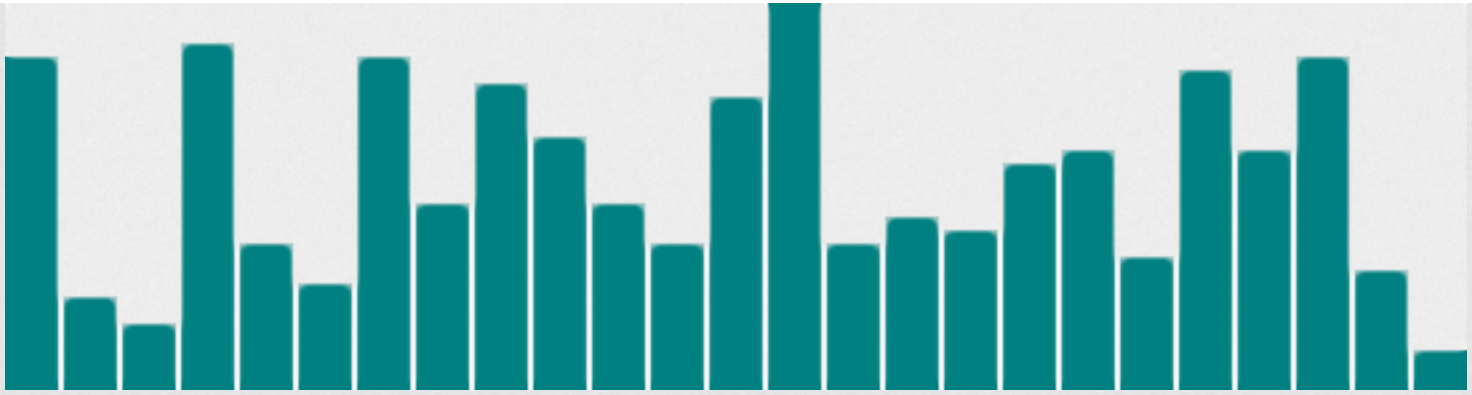
Está bien, pero los datos en la vida real nunca son así. Cambiemos nuestros datos

```
var dataset = [ 25, 7, 5, 26, 11 ];
```



Tampoco vamos a limitarnos a cinco datos. Podemos añadir más!

```
var dataset = [ 25, 7, 5, 26, 11, 8, 25, 14, 23, 19,
                14, 11, 22, 29, 11, 13, 12, 17, 18, 10,
                24, 18, 25, 9, 3 ];
```



25 datos en vez de cinco! Cómo es que D3 expande automáticamente nuestro diagrama si es necesario?

```
d3.select("body").selectAll("div")
.data(dataset)  // <-- Acá está la respuesta!
.enter()
.append("div")
.attr("class", "bar")
.style("height", function(d) {
    var barHeight = d * 5;
    return barHeight + "px";
});
```

Si se le pasan diez datos a la función `data()`, ésta recorrerá la cadena diez veces. Si se le pasan un millón de datos, la recorrerá un millón de veces (solo tiene que tener paciencia).

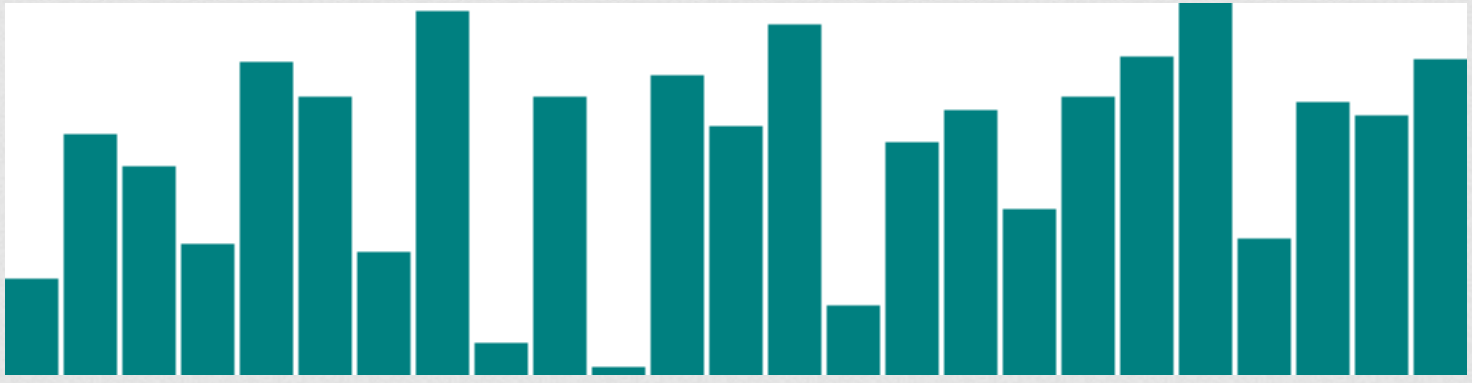
Este es el poder de la función `data()`, la cual es lo suficientemente inteligente para recorrer por completo cualquier conjunto de datos que se le dé y ejecutar cada una de las funciones siguientes en la cadena. Esto lo hace mientras actualiza el contexto dentro del cual opera cada función, de tal manera que `d` siempre se refiere al dato actual, o aquel dónde esté el ciclo (loop) en ese momento.

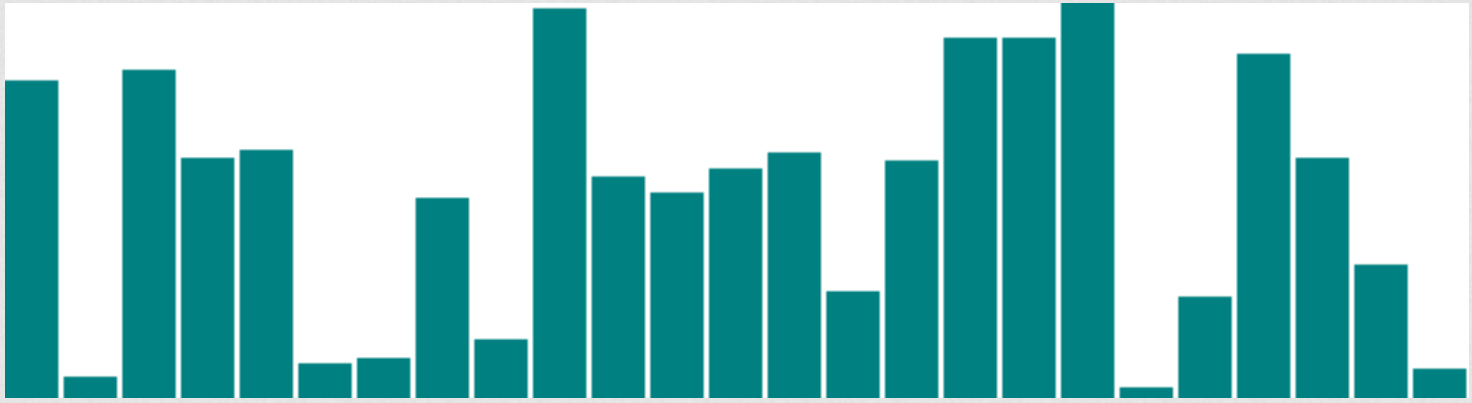
Puede que esto sea demasiada información. Si no la entiende del todo en este momento, pronto la entenderá. Le recomiendo que guarde el código fuente de las páginas HTML de ejemplo anteriores, cambie los valores del conjunto de datos y revise cómo cambia el diagrama de barras.

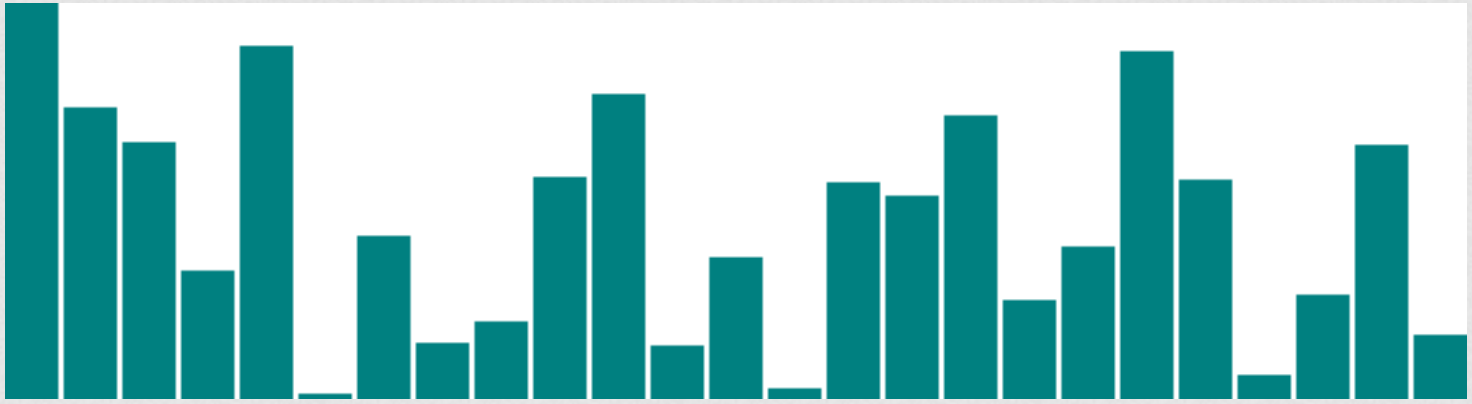
Recuerde, los datos son lo que generan la visualización y no al contrario.

Datos Aleatorios

Algunas veces es divertido generar conjuntos con datos aleatorios, pero efectos de chequeo o solamente por interés en temas técnicos. Eso es lo que he hecho acá. Puede ver cómo cambia la gráfica cada vez que se refresca la página.







Si revisa el código fuente, podrá ver: var dataset = []; //Initialize empty array for (var i = 0; i < 25; i++) { //Loop 25 times var newNumber = Math.random() * 30; //New random number (0-30) dataset.push(newNumber); //Add new number to array }

Este código no utiliza ningún método de D3, es solamente JavaScript. Sin entrar en mucho detalle, este código:

1. Crea un arreglo vacío denominado `dataset`.

2. Inicia un ciclo (loop) `for`, que se ejecuta 25 veces.
3. Por cada ciclo, se genera un número aleatorio entre 0 y 30.
4. El nuevo número se añade al arreglo `dataset`. (`push()` es un método de arreglos que añade un nuevo valor al final del arreglo.)

Solamente por jugar, abra la consola de JavaScript y escriba `console.log(dataset)`. Podrá ver todo el arreglo con 25 valores generados aleatoriamente.

```
> console.log(dataset)
  [14.793717765714973, 21.65710132336244, 22.01914135599509, 10.693866850342602,
  8.197558452375233, 8.327909619547427, 9.349913026671857, 6.715130957309157,
  20.352523955516517, 20.892786516342312, 18.432767554186285, 7.062793713994324,
  11.519823116250336, 8.91862049465999, 5.422192756086588, 8.956057007890195,
  13.239774140529335, 24.165618284605443, 14.453229457139969, 27.792113937903196,
  2.717762708198279, 12.752952876035124, 1.7288982309401035, 21.01240729680285,
  26.07524922117591]
```

Puede ver que todos los números son decimales o puntos flotantes (14.793717765714973) y no números reales o enteros (14) como usamos inicialmente. Para efectos de este ejemplo, esto basta, pero si alguna vez necesita números enteros, puede usar el método de JavaScript `Math.round()`. Por ejemplo, se puede envolver el generador de números aleatorios de esta línea:

```
var newNumber = Math.random() * 30;
así:
var newNumber = Math.round(Math.random() * 30);
```

Pruébelo aquí y use la consola para verificar que los números se hayan redondeado a números enteros.

Ahora vamos a expandir las posibilidades de visualización con SVG.

Siguiente: Introducción a SVG –>

Este tutorial fue traducido por Gabriel Coch